Introduction

Description of the motivation for undertaking the task of creating selective transparent headphones.

The advancement of signal processing technologies has consistently improved the enjoyment of listening to music. For example, noise cancelling techniques have been widely applied to headphones that, in addition to physical noise isolation, provide further reduction in noise.

We believe, however, that it is sometimes important for music listeners with a pair of good noise isolating headphone to be aware of certain surrounding signals. For example, when people are crossing the street while listening to music, they really want to hear car horns to be warned of possible danger. We also think that human speech is important, and that people may want to hear others talking to them while they are listening to music.

Based on the assumption above, we propose to build a selective transparent headphone that propagates certain surrounding sound signals and suppresses all other signals. In this project, we explore possibility to select only speech signal to propagate and silence other surrounding sounds.

The following sections are organized as follows. The second section provides an overview of the project, defining the problem and explaining high level system implementation. The third to the fifth sections explain in detail the blind source separation algorithm and binary artificial neural network, respectively. The following sections discuss experiment results, and suggest next steps.

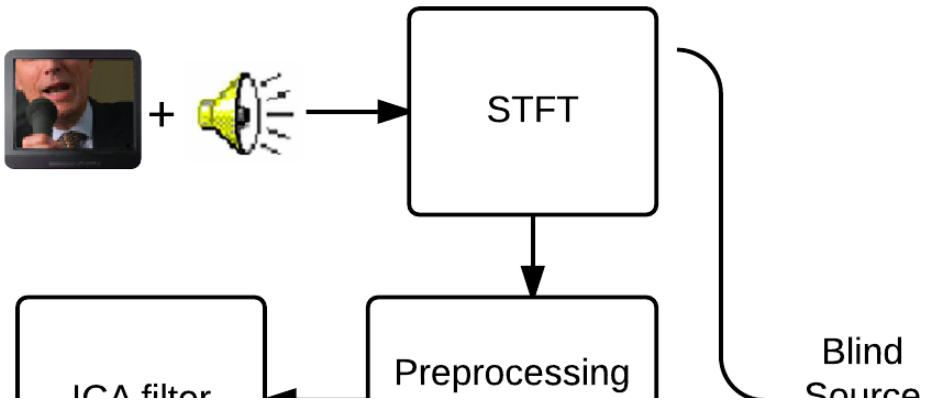
Project Overview
Overview of the steps followed to create the system.

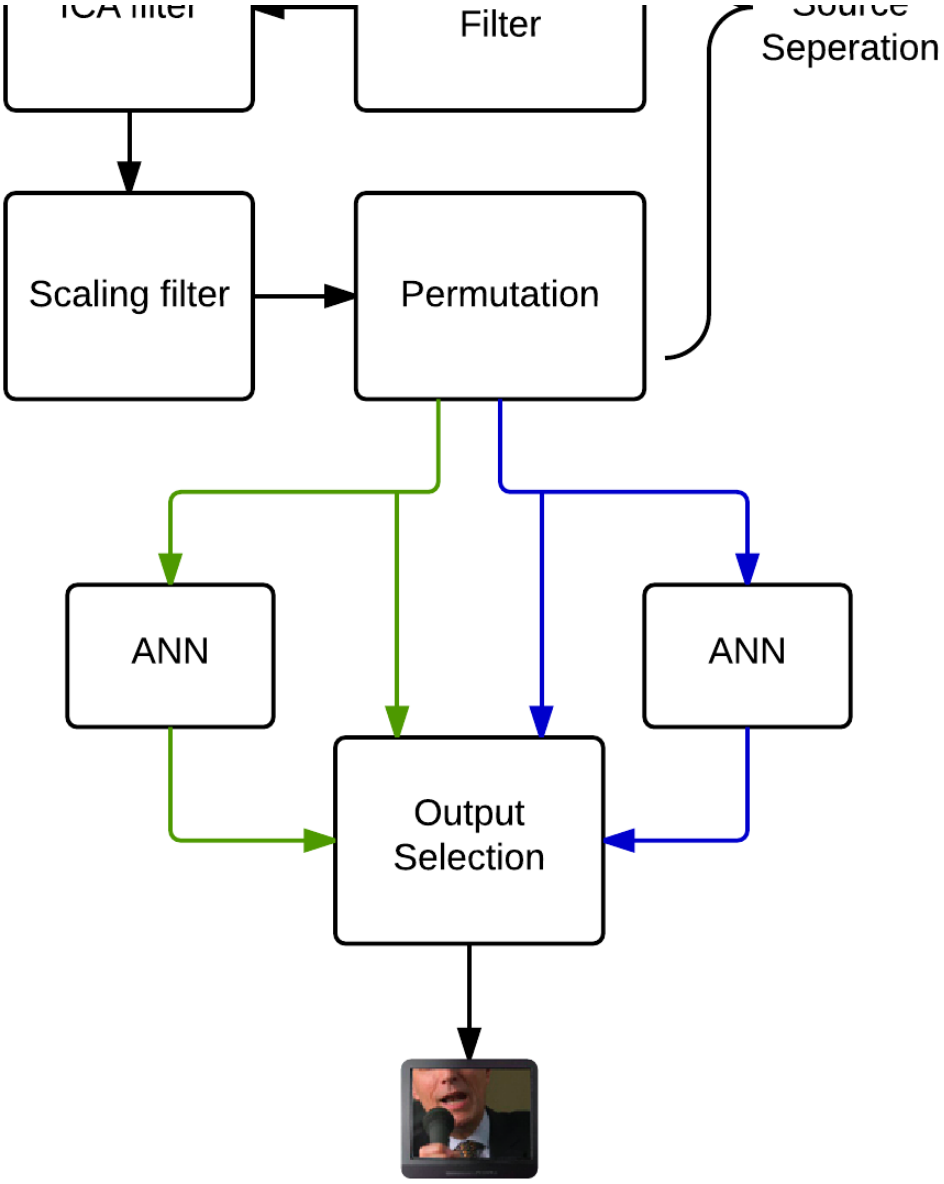
Problem Definition

There are a number of sound signals that are of importance in the environment. In this project, we identify human speech as an important signal, that we would like to focus on. We would like to separate the environment sound signals into a signal containing only human speech and a signal containing all other speech. By making the above simplifications, the problem is reduced to finding the speech content in the surrounding environment, and forwarding the speech content to the listener while suppressing other signals. If there are no speech signals, or speech signals are weak compared to other signals, all sounds from the environment will be attenuated.

System Implementation

We approach the problem stated above by first separating the source signals into human speech content and non-human speech with a blind source separation algorithm. After that, a classification algorithm determines which signal contains human speech, if any, and outputs that signal. A detailed overview is shown below with the system block diagram in Figure 1.





The audio input, a mixture of human speech and some other noise such as instrumental music, is passed to the blind source separation block. Inside this block, short time Fourier transform is first performed to change the time domain input signal into frequency domain, since all the other operations on the signals reside in frequency domain. After that a preprocessing filter is applied, which cleans the input signal by removing reflection and ambient noises. Then for each frequency, the independent component analysis algorithm separates the signal into two parts such that independence between the two signals is maximized. Scaling and

permutation filters minimizes distortions caused by using the independent component analysis method. The blind source separation process outputs two signals, where at most, one signal contains speech.

We then implement a binary artificial neural network (ANN) for classification.The two identical ANN take the two signals separated from the source as inputs respectively and outputs a weight for each signal. We train the neural network in the way that a signal containing more human speech will output a higher weight. Based on the weights of the separated signals, the output selection multiplexer chooses the proper signal to output. That is, if one signal, referred to as A, has a higher weight than the other signal, referred to as B, and its weight is above a certain threshold (0.5), the multiplexer outputs signal A. If both the weights of signal A and signal B are below that threshold, then the multiplexer outputs nothing. We set a threshold to deal with situation where both signal contain no human speech.

Background on Independent Component Analysis
Describes the motivation and background behind using independent component analysis as the basis of the blind source separation algorithm used in this endeavor.

The blind source separation algorithm, we employed, is based around independent component analysis, which is explained in more detail below. The goal is to recover independent sources given only sensor observations that are linear mixtures of independent source signals. We assume that the source signals are statistically independent and non-Gaussian.

ICA works by solving the following model.

$$x = As$$

Where vector **x** contains *n* observed signals, vector **s** contains independent sources that comprise **x**, and matrix **A** denotes the mixing matrix. It is assumed that the mixing matrix **A** and the source vector **x** are both unknown. ICA algorithm thus produces the best estimation for both the mixing matrix and source vector.

The first step in ICA is usually whitening (sphering) the data, which removes any correlations in the data, i.e. the signals are forced to be uncorrelated. Putting the words in mathematical terms, we seek a linear transformation **V** such that when **y = Vx** we now have **E[yy'] = I**.

After sphering, the separated signals can be found by an orthogonal transformation of the whitened signals **y** (this is simply a rotation of the joint density). The appropriate rotation is sought by maximizing the non-normality of the marginal densities. This is because of the fact that a linear mixture of independent random variables is necessarily more Gaussian than the original variables.

Although in theory ICA algorithm can perfectly separate source signals from observed signals, provided that sources are statistically independent and non-Gaussian, most of the algorithms do not work quite well in real world scenarios. For example, ICA does not typically deal with reflection; that is, if mixed signal is observed in a small room with reverberation, ICA algorithms such as FastICA hardly identifies individual components from

the observed signal. Therefore, we need a more robust blind source separation algorithm, which is discussed in the following section.

Blind Source Separation
Describes the method to separate an arbitrary signal into its independent components for the application of creating a selective transparent headphone.

Background

When separating mixtures of instantaneously mixed signals, independent component analysis works very well, but this ideal situation is rarely found in the real-world. In practical applications, the environment distorts audio signals by adding echoes, reflections, and ambient noise. Additionally independent component analysis, in its purest form, assumes that source signals do not have any propagation delay, which is an assumption that cannot be applied in this case. Recording sources from two microphones placed in different locations will inevitably introduce propagation delays, so the blind source separation method used must also consider this issue.

To solve problems detailed above, the blind source separation problem will be redefined in the time-frequency domain. By taking the short-time Fourier transform (STFT) of the audio inputs, we can represent the inputs as the following.

$$x(\omega, t) = [X_1(\omega, t), X_2(\omega, t), ..., X_M(\omega, t)]^T$$

Xi(ω,t) refers to the input observed at the *i*th microphone observed at time *t*. The *T* symbol refers to the transpose, so in this case, the input sources are represented along the rows. For our application, we will assume that the number of observed inputs and the number of separated sources are both equal to the constant *M*. We can, now, formulate our blind source separation problem as the following.

$$x(\omega, t) = A(\omega)s(\omega, t) + n(\omega, t)$$

In this case, **s(ω,t)**, refers to the vector of source signals observed at frequency ω and at time *t*. The second term, **n(ω,t)**, represents any type of noise or distortions that may be present in the observed signals (noise, reflections, etc.). The blind source separation method attempts to solve for the mixing matrix, **A(ω)**, which can be represented as the following.

$$A_{i,j}(\omega) = H_{i,j}(\omega)e^{-j\omega\tau_{i,j}}$$

Here, **Hi,j(ω)**, refers to the transfer function of the *j*th source to the *i*th microphone. Additionally, **τi,j**, refers to the propagation delay from the *j*th source to the *i*th microphone.

The system, for which our problem is defined on, is now redefined from an instantaneous mixture of signals to a convolutional mixture in the time-domain of the following form.

$$x(t) = \hat{a}(t) * s(t)$$

This formulation allows for a solution which considers the distortion, **n(ω,t)**, added by the environment and considers the inherent propagation delay that is inherently present in our application.

Since the new formulation of the blind source separation problem is a convolutional system in the time-domain, we will solve the separation problem in the time-frequency domain formulation shown in Figure 2. Given our approach outlined above, we can solve for the complete frequency-domain separation filter shown below.

$$G(\omega) = P(\omega)\tilde{B}_m^+(\omega)B(\omega)$$

B(ω) is the separation filter of the system. Applying this filter will separate the spectrum of the observed signal into the spectra of the source signals

observed at frequency, ω. As stated above, by reformulating the problem into the frequency domain, the convolutional mixture becomes transformed into an instantaneous mixture. As such, independent component analysis can be applied to separate the source signals at each frequency. However, in doing so, we introduce other problems, inherent to the independent component analysis method, that the other filters attempt to resolve.

The following sections will explain the three filters in detail.

- **B(ω)**, the separation filter.
- **Bm+**, the normalization filter used to solve the scaling problem that arises after applying independent component analysis.
- **P(ω)**, the permutation filter used to solve the frequency distortion problem that arises after applying independent component analysis.

Separation Filter

The separation filter, shown in the previous section can be broken down into the following components.

$$B(\omega) = U(\omega)W(\omega)$$

W(ω) is the preprocessing filter used to reduce reflections and ambient noise by using the subspace method. **U(ω)** is the filter obtained by applying independent component analysis after preprocessing.

I. Subspace Method

The spatial correlation, or autocorrelation, matrix at frequency ω is defined below.

$$R(\omega) = E[x(\omega, t)x^H(\omega, t)]$$

Given that there are *M* inputs and *M* sources, the resulting matrix will be *M*×*M*. This matrix can be rewritten in the following way.

$$R(\omega) = AQA^H + K$$

The spatial correlation matrix.

$$K = E[n(t)n^H(t)]$$

The noise correlation matrix.

$$Q = E[s(t)s^H(t)]$$

The sources' cross-spectrum matrix.

In other words, **K** is the correlation matrix of **n(t)** and **Q** is the cross-spectrum matrix of the sources. However, since the source signals, **s(t)**, are unknown at this point, this equation cannot be directly solved. Instead, we can represent the generalized eigenvalue decomposition of the spatial correlation matrix in the following manner.

$$R = KE\Lambda E^{-1}$$

E refers is the eigenvector matrix, **E = [e1,e2, …,eM]**, and **Λ = diag(λ1,λ2,…,λM)** refers to the eigenvalues of **R**. Since **K**, the noise correlation matrix, cannot be directly observed apart from the source signals, we will assume that **K = I**, which will evenly distribute the reflections and ambient noise induced by the environment among the estimated sources. As such, the eigenvalue decomposition of **R** can be rewritten as the following.

$$R = E\Lambda E^{-1}$$

The final preprocessing filter uses the eigevector and eigenvalue matrices resulting from the standard eigenvalue decomposition of the spatial correlation matrix and is shown below.

$$W = \Lambda^{-1/2} E^{H}$$

II. Independent Component Analysis

By applying the preprocessing filter obtained from the subspace method, our observed signals are now of the following form.

$$y(\omega, t) = W(\omega)x(\omega, t)$$

To estimate the source signals from the preprocessed input signals, we can use independent component analysis to solve the linear system displayed below, for the filter **U(ω)**.

$$y(\omega, t) = U^{-1}(\omega)\hat{s}(\omega, t)$$

Now that the separation filter at each frequency has been found, the problems that arise from the separation process must be addressed.

Scaling

The scaling problem that results from applying independent component analysis can be resolved using the filter shown below.

$$\tilde{B}_m^+(\omega) = diag[B_{m,1}^+, B_{m,2}^+, ..., B_{m,M}^+]$$

Bm,n+(ω) refers to the *(m,n)*th entry in **B+(ω)**, which is the pseudo-inverse of **B(ω)** (regular inverse also works here since we assume that the number of separated sources and the number of inputs are both *M*, so that the resulting separation matrix is square). Additionally, *m* refers to an arbitrary row or microphone in **B+(ω)**. By applying this matrix, each signal, *i*, is amplified by the component of signal *i* observed at microphone or input *m*. Essentially, this filter amplifies the frequency components of the separated source signals so that the waveforms of the resulting sources will be distinguishable and audible.

Permutation

A second critical problem that arises after applying independent component analysis is that the order in which the source signals are returned is unknown. Since independent component analysis is applied at each frequency, we must find the permutation of components that has the highest chance of being the correct permutation to reconstruct the source signals and minimize the amount of frequency distortion caused by independent component analysis.

We define the permutation matrix, **P**, as the following.

$$Z(\omega) = B^{+}(\omega)$$

$$\bar{Z}^{T}(\omega) = PZ^{T}(\omega)$$

$$\bar{Z}(\omega) = [\bar{z_1}(\omega), \bar{z_2}(\omega), ..., \bar{z_M}(\omega)]$$

The matrix, **P**, exchanges the column vectors of **Z(ω)** to get different permutations. We define the cosine of *θn* between the two vectors **zn(ω)** and **zn(ω0)**, where *ω0* is the reference frequency is as follows.

$$cos(\theta_n) = \frac{\bar{z}_n^H(\omega)\bar{z}_n(\omega_0)}{||\bar{z}_n(\omega)|| \cdot ||\bar{z}_n^H(\omega_0)||}$$

The permutation is, then, determined by the following.

$$P = \arg\max_P F(P)$$

The cost function, **F(P)**, used above, is defined as follows.

$$F(P) = \frac{1}{D}\sum_{n=1}^{M} cos(\theta_n)$$

This compares the inverse vector filter at each frequency with a filter at a reference frequency, *ω0*. However, if we use the same reference frequency to find the permutation at every other frequency, then the problem arises if the filter at the reference frequency has the incorrect permutation. In order to minimize this error, a new reference frequency is chosen after the permutations of filters at **K** frequencies are found. As such, the frequency range of a reference frequency, *ω0*, is as follows.

$$\omega_0 = \omega - k \cdot \Delta\omega, \forall k = 1, ..., K$$

Background on Artificial Neural Network
This module talks about the backgrounds on artificial neural network

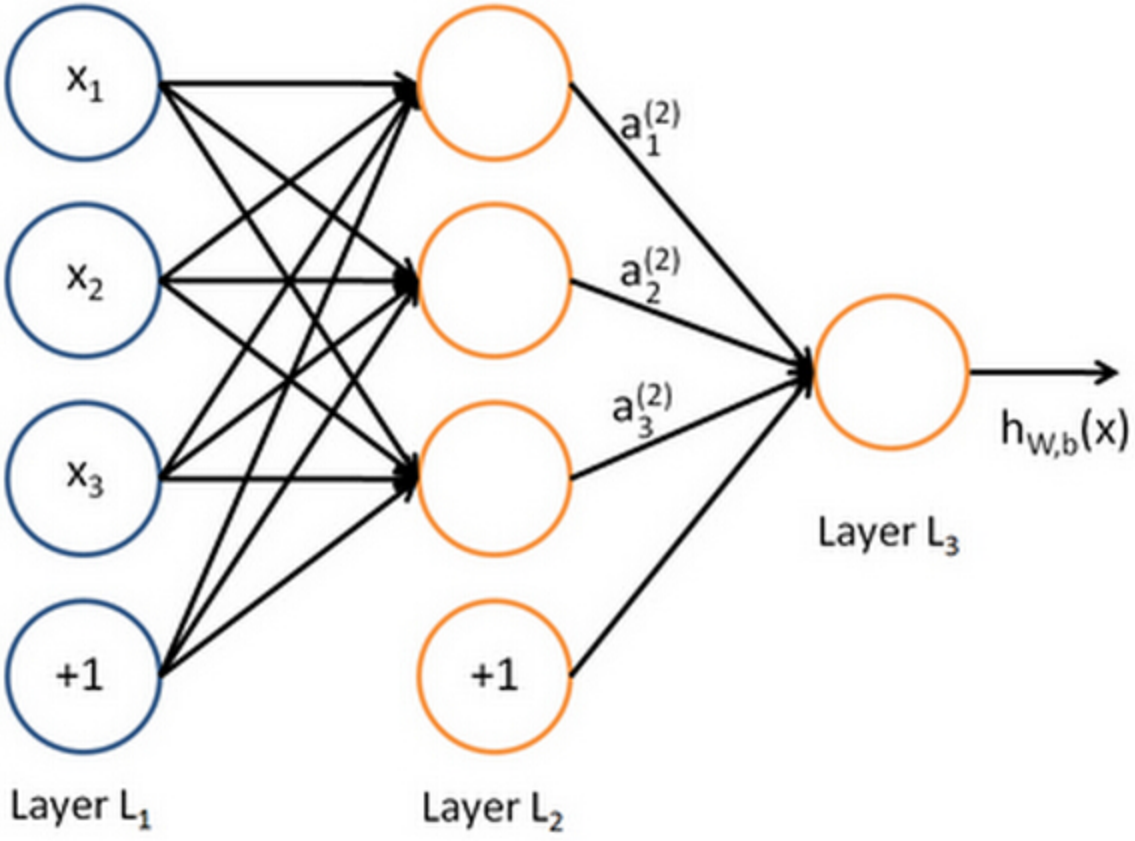
Motivation:

As the Blind Source Separation algorithm randomly assigns the two independent components into two output vectors, we need the artificial neural network to determine which of the two signals, if any, contains human speech.

Background on Our Chosen Neural Network Model:

Our Neural Networks Model:

A simplified schematics of our neural network looks like the following:



Neural Network Model

Our artificial neural network consists of one input layer (Layer L1), one hidden layer (Layer L2), and one output layer (Layer L3). The circles labeled “+1” are called the bias units, and corresponds to the intercept term.

Our neural network has parameters $(W,b) = (W^{(1)},b^{(1)},W^{(2)},b^{(2)})$, where we write $W^{(l)}_{ij}$ to denote the weight associated with the connection between unit j in layer l, and unit i in layer l+1. Also, $b^{(l)}_i$ is the bias associated with unit i in layer l+1. We also use s_l to denote the number of nodes in layer l (not counting the bias unit).

We will write $a^{(l)}_i$ to denote the activation of unit i in layer l. For l=1, we also use $a^{(1)}_i = x_i$ to denote the i-th input. Given a fixed setting of the parameters W,b, our neural network defines a hypothesis $h_{W,b}(x)$ that outputs a real number. Specifically, the computation that this neural network represents is given by:

$$
\begin{aligned}
a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\
a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\
a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\
h_{W,b}(x) &= a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})
\end{aligned}
$$

Let $z^{(l)}_i$ denote the total weighted sum of inputs to unit i in layer l, including the bias unit. For instance,

$$z_i^{(2)} = \sum_{j=1}^{n} W_{ij}^{(1)}x_j + b_i^{(1)})$$

, so that

$$a_i^{(l)} = f(z_i^{(l)})$$

.

If we extend the activation function f(.) to apply to vectors in an element-wise fashion, we can write the above equations more compactly as:

$$
\begin{aligned}
z^{(2)} &= W^{(1)}x + b^{(1)} \\
a^{(2)} &= f(z^{(2)}) \\
z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\
h_{W,b}(x) &= a^{(3)} = f(z^{(3)})
\end{aligned}
$$

The above equations represents the forward propagation step. More generally, as we use $a^{(1)} = x$ to denote the values from the input layer, then given layer l's activations $a^{(l)}$, we can compute layer l+1's activations $a^{(l+1)}$ as:

$$
\begin{aligned}
z^{(l+1)} &= W^{(l)}a^{(l)} + b^{(l)} \\
a^{(l+1)} &= f(z^{(l+1)})
\end{aligned}
$$

Backpropagation Algorithm:

Suppose we have a fixed training set

$$\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$$

of m training examples, the overall cost function is:

$$
\begin{aligned}
J(W,b) &= \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 \\
&= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}(x^{(i)}) - y^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2
\end{aligned}
$$

The first term in the cost function is an average sum-of-squares error term. The second term is the regularization term which is used to prevent

overfitting. The weight decay parameter λ controls the relative importance of the cost term and the regularization term.

Our goal is to minimize the cost function J(W,b) as a function of W and b. To begin training our neural network, we will initialize each weight $W^{(l)}_{ij}$ and each $b^{(l)}_i$ to a small value near zero. It is important to initialize the parameters randomly for the purpose of symmetry breaking.

In order to perform gradient descent to minimize the cost function, we need to compute the derivatives of the cost function with respect to $W^{(l)}_{ij}$ and $b^{(l)}_i$ respectively:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b) = \left[\frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x^{(i)},y^{(i)}) \right] + \lambda W_{ij}^{(l)}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W,b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial b_i^{(l)}} J(W,b;x^{(i)},y^{(i)})$$

The intuition behind the backpropagation algorithm is as follows. Given a training example (x,y), we will first run a “forward propagation” to compute all the activations, including the output value of the hypothesis $h_{W,b}(x)$. Then for each node i in layer l, we compute an error term $\delta^{(l)}_i$ that measures how much that node was responsible for any errors in the output. For an output node, we can directly measure the difference between the network’s hypothesis and the true value, and use that to define the error term for the output layer.

Here is the implementation of the backpropagation algorithm in MatLab:

1. Perform a feedforward propagation, computing the activations for hidden layer L2 and output layer L3.
2. For the output layer (layer n_l), set

$$\delta^{(n_l)} = -(y - a^{(n_l)}) \bullet f'(z^{(n_l)})$$

1. For the hidden layer, set

$$\delta^{(l)} = \left((W^{(l)})^T \delta^{(l+1)}\right) \bullet f'(z^{(l)})$$

1. Compute the desired partial derivative of the cost function with respect to W and b

$$\nabla_{W^{(l)}} J(W, b; x, y) = \delta^{(l+1)} (a^{(l)})^T,$$
$$\nabla_{b^{(l)}} J(W, b; x, y) = \delta^{(l+1)}.$$

Implementation note: in step 2 and 3 above, we need to compute f'($z^{(l)}_i$) for each value of i. as our f(z) is the sigmoid function and we already have $a^{(l)}_i$ computed during the feedforward propagation process. Thus using the expression for f'(z), we can compute this as

$$f'(z_i^{(l)}) = a_i^{(l)}(1 - a_i^{(l)}).$$

.

Below is the pseudo code of the gradient descent algorithm:

Note: $\Delta W^{(l)}$ is a matrix of the same dimension as $W^{(l)}$ and $\Delta b^{(l)}$ is a vector of the same dimension as $b^{(l)}$. one iteration of the gradient descent as follows:

1. Set $\Delta W^{(l)} := 0$, $\Delta b^{(l)} := 0$ for all l.
2. For i from 1 to m,

a. Use backpropagation to compute

$$\nabla_{W^{(l)}} J(W, b; x, y) \text{ and}$$
$$\nabla_{b^{(l)}} J(W, b; x, y).$$

a. Set

$$\Delta W^{(l)} := \Delta W^{(l)} + \nabla_{W^{(l)}} J(W, b; x, y).$$

b. Set

$$\Delta b^{(l)} := \Delta b^{(l)} + \nabla_{b^{(l)}} J(W, b; x, y).$$

1. Update the parameters:

$$W^{(l)} = W^{(l)} - \alpha \left[\left(\frac{1}{m} \Delta W^{(l)} \right) + \lambda W^{(l)} \right]$$
$$b^{(l)} = b^{(l)} - \alpha \left[\frac{1}{m} \Delta b^{(l)} \right]$$

We can now repeat the gradient descent steps to reduce our cost function J(W,b).

Implementation and Experimental Results
This is our implementation of neural network and Blind Source Separation and their respective results

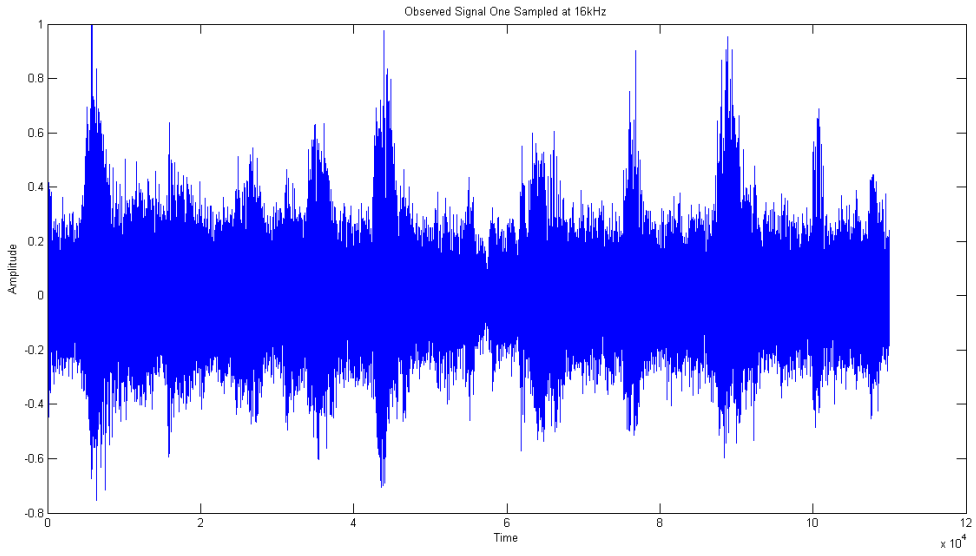
Implementation of Blind Source Separation:

When implementing the Blind Source Separation algorithm, we chose the following parameters,

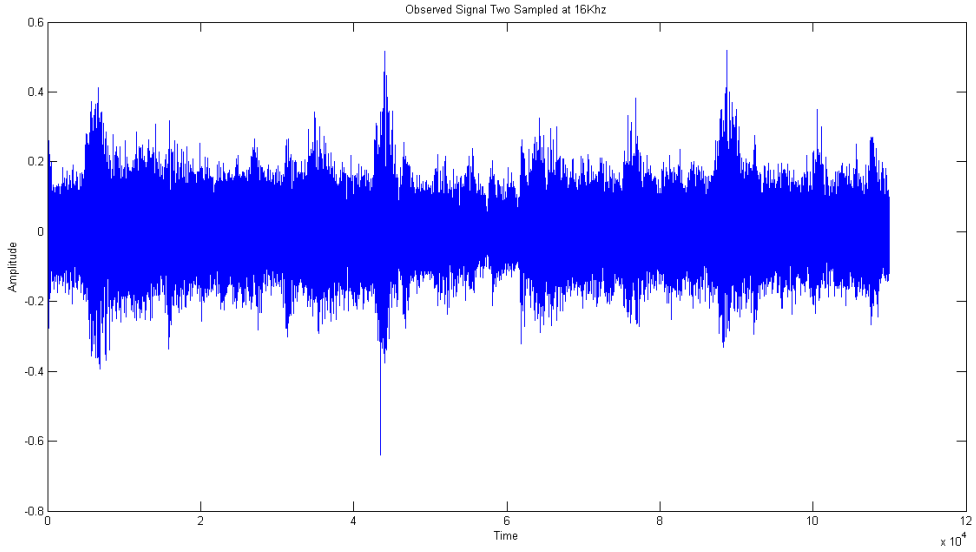
	Parameters
Sampling Rate	16 kHz
Length of STFT	512
Number of FFT Points	4096
Overlap	492
Number of inputs	2
Permutation Reference Range	5

Input Parameter Into The System

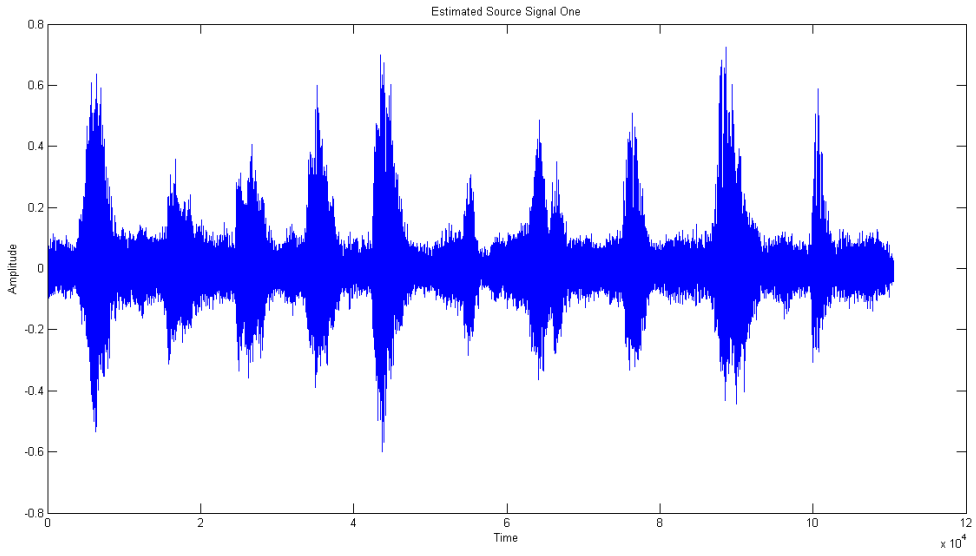
Result of our Blind Source Separation:



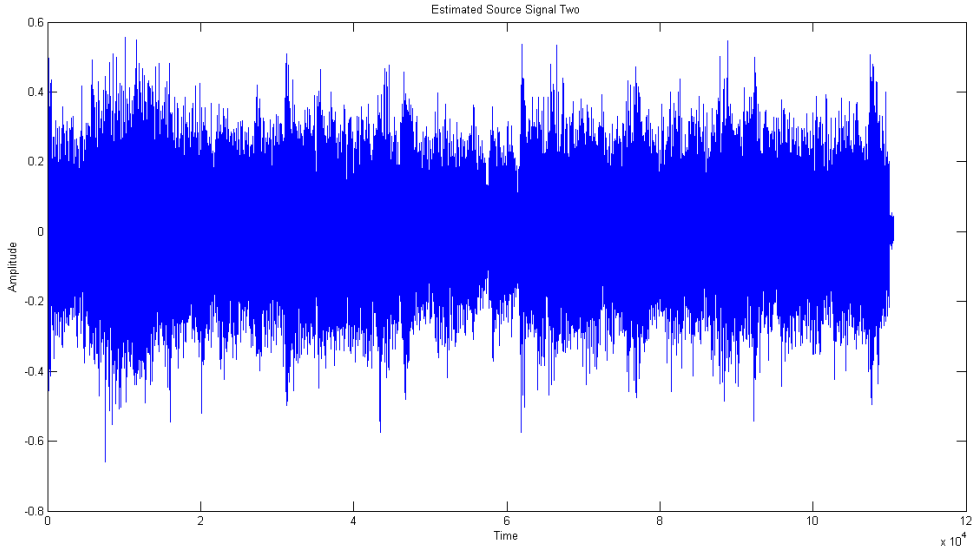
Mixed Signal Observed at 1st Microphone



Mixed Signal Observed at 2nd Microphone



Estimated Source Signal One-human speech



Estimated Source Signal Two-Instrumental Music

Figure 2 and Figure 3 show the mixed signals observed at the 1st microphone and the 2nd microphone respectively. The mixed signal is a

recording of a person counting from zero to ten with instrumental music playing in the background. The intermittent spikes in the mixed signal represents the person counting while the continuous, small-amplitude signal represents the instrumental music.

After applying the Blind Source Separation algorithm, we obtain two independent signals shown in Figure 4 and Figure 5 respectively. These two signals are very distinct from each other. The output signal 1 shown in Figure 4 consists mainly of the person counting whereas the output signal 2 shown in Figure 5 is primarily the instrumental music.

Implementation of Neural Network:

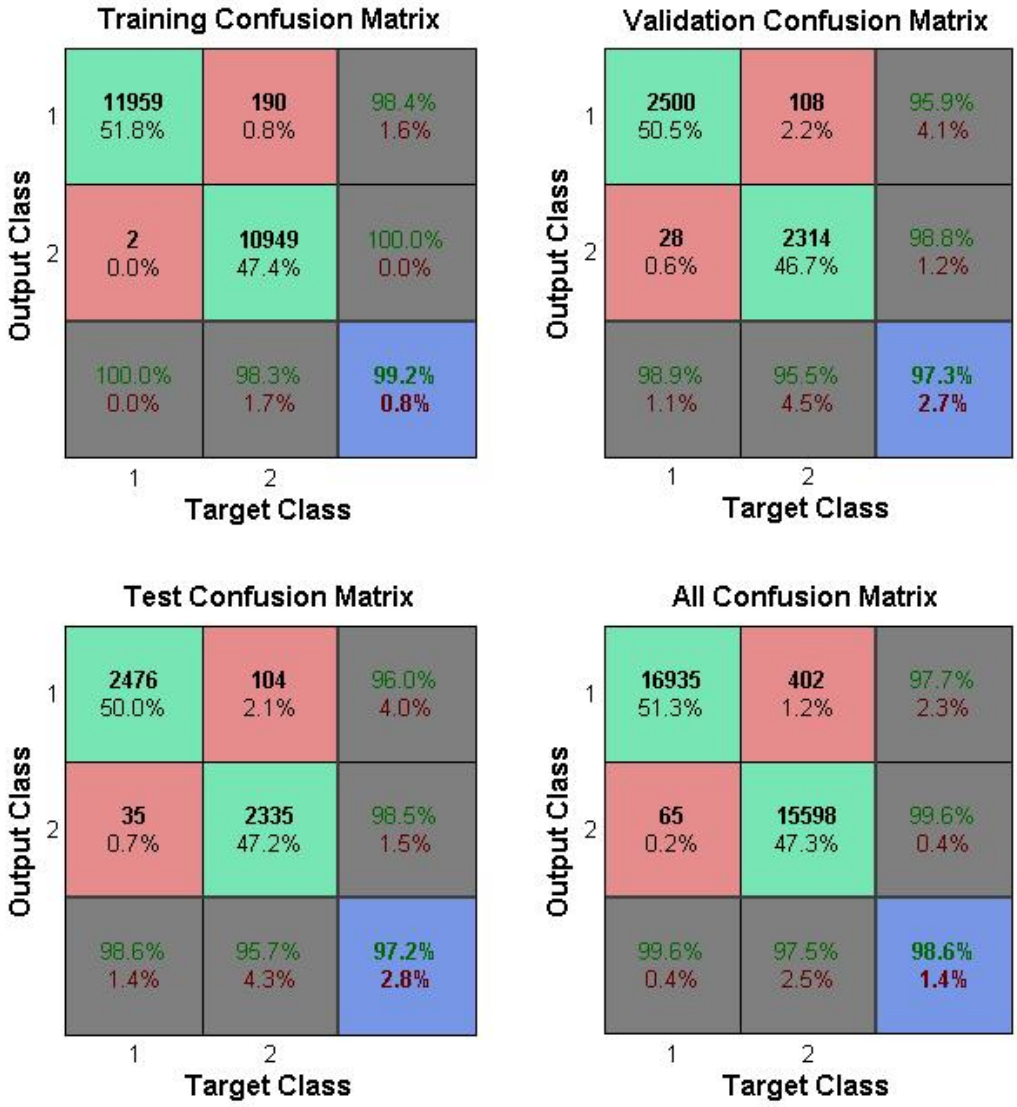
Since we need our neural network to perform binary classification, we used 512 nodes for the input layer, 25 nodes for the hidden layer, and 2 nodes for the output layer.

We used 10,000 positive training examples and 10,000 negative training examples to train our neural network. The positive training examples mainly consist of audio file containing examples of human speech such as news broadcasting. The negative training examples mainly consist of examples of non-human speech such as instrumental music.

After taking 512 point Short-Time Fourier Transform, we place our training sets into a 20,000-by-512 matrix where each row represents a single training example. Then we used the backpropagation and gradient descent algorithm described in the section above to train our neural network. We used 70% of the training examples for training, 15% of the training examples for validation, and another 15% for testing.

Result of Our Training of Neural Network:

After employing backpropagation algorithm and gradient descent algorithm to train our neural network, we get the following test results:



Accuracy of Neural Network

As our neural network is a binary classification model, our output is a vector containing two entries. The first entry represents the probability of the input signal being a human speech and the second entry represents the probability of the input signal being a non-human speech. The two entries sum to one. The overall performance of our neural network is shown in the

All Confusion Matrix. The red block showing 0.2% means o.2% of the class 1 examples (human speech) has been incorrectly identified as class 1 (non-human speech). Similarly, the red block showing 1.2% mean 1.2% of the examples of class 2 (non-human speech) has been incorrectly identified as class 1 (human speech). The blue block shows that our neural network is able to distinguish between human speech and non-human speech with an accuracy of 98.6%.

Conclusion and Next Steps
This is our conclusion

Conclusion

From our experimental results, we can see that our Blind Source Separation algorithm is able to separate the two independent components of the mixed signal very well. After the separation, the neural network can distinguish between human speech and non-human sound with an accuracy of 98.6%. Consequently, our entire system can successfully output the human speech from a linear mixture of sound signals.

Next Steps

Implement the system in real-time. Our design currently works in the Matlab environment. It is implemented to deal with audio that is already recorded, but it does not deal with real-time audio streams. We eventually would like the system to work in real-time, separating the sources, forward speech, if any, and suppress other signals instantly so that people can benefit from this system. This implies improving the efficiency of the algorithms, and making use of Matlab SimuLink to explore the possibility of real-time implementation of our system.

Additionally, we would like to explore the possibility to separate several sources in addition to human speech. As mentioned in the introduction, other signals from the surrounding might also be important, such as a car's horn. We can improve the separation algorithm and train the neural network so that the system can recognize not only human speech, but also other important signals from the environment. With this implementation, users can have the option to forward in the signals they want. In order to achieve that, one needs to increase the number of microphones to increase the number of observed signals, and re-train the neural network so that it recognizes different types of sound signals.

Team Members and Acknowledgements

Team Members

Contact Information

Wang, Zichao: zichao.wang@rice.edu

Xia, Stephen: stephen.xia@rice.edu

Yao, Tianyi: tianyi.yao@rice.edu

Zhao, Shengjia: shengjia.zhao@rice.edu

Work Distribution

Blind Source Separation: Zichao and Stephen

Neural Network: Tianyi and Shengjia

Poster and Presentation: Zichao, Stephen, Tianyi, Shengjia

Documentation: Zichao, Stephen, Tianyi, Shengjia

Acknowledgements

We would like to thank Dr. Mohammad Golbabaee for mentoring and providing us with many insights throughout our project. Additionally, we would like to extend our thanks to Professor Richard Baraniuk and the Rice University Electrical and Computer Engineering Department for giving us the opportunity to partake in this endeavor.

References

Asano, Futoshi, et al. "Combined approach of array processing and independent component analysis for blind separation of acoustic signals."Speech and Audio Processing, IEEE Transactions on 11.3 (2003): 204-215.

"Backpropagation Algorithm." Unsupervised Feature Learning and Deep Learning. Stanford University, 26 Feb. 2011. Web. 16 Dec. 2014.

Hyvärinen, Aapo, and Erkki Oja. "Independent component analysis: algorithms and applications." Neural networks 13.4 (2000): 411-430.

"Neural Networks." Unsupervised Feature Learning and Deep Learning. Stanford University, 26 Feb. 2011. Web. 16 Dec. 2014.

Widrow, Bernard, et al. "Adaptive noise cancelling: Principles and applications." Proceedings of the IEEE 63.12 (1975): 1692-1716.

Poster
This is our poster

Selective Transparent Headphone

Zichao Wang, Tianyi Yao, Stephen Xia, Shengjia Zhao

Rice University Department of Electrical and Computer Engineering

RICE ece redefining limits

Objective

Explore ways to build a selective transparent headphone that propagates speech signal and attenuates all other types of signals.

Project Overview

- Since our objective is to blindly separate an input signal into its principal/independent components, we will use independent component analysis to separate the signal into its components.
- Then, we will utilize a neural network to select which output signal, if any, contains human speech, and forward it to the output.

Independent Component Analysis

- Problem formulation:
  - *y = input signals (sound mixtures)*
  - *s = original sources*
- ICA attempts to find the independent sources (individual sounds) that comprise an input signal by solving the following formulation for A^{-1}

$$y = As$$

- *A* is the mixing matrix used to combine the original sources, *s*, to obtain the observed input signals, *y*.

Blind Source Separation (BSS)

- Transform *M* microphone inputs into frequency domain by taking the short-time Fourier transform (STFT).

$$x(\omega,t) = [X_1(\omega,t),\ldots,X_M(\omega,t)]^T$$

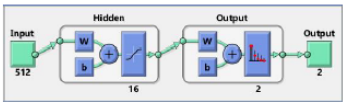
- Find subspace filter to reduce reflections and ambient sound.

$$W(\omega) = \Lambda_z^{-1/2} E_z^H$$

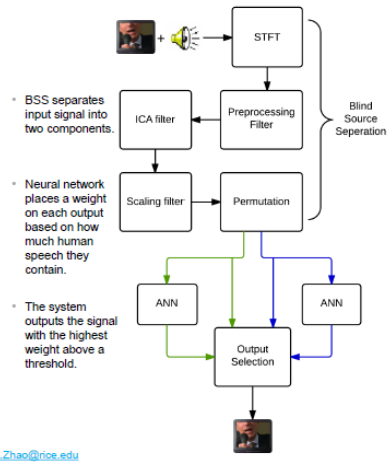
- Find ICA filter matrix, $U(\omega)$, for all sampled frequencies.
- Separation filter: $B(\omega) = U(\omega)W(\omega)$
- Scaling filter: $\tilde{B}_m^*(\omega) = diag[B_{m,1}^*,\ldots,B_{m,D}^*]$
- Permutation filter: $F(P) = \frac{1}{D}\sum_{\omega}^{\pi}\cos\theta_\omega$
- Final filter: $F(\omega) = P(\omega)\tilde{B}_m^*(\omega)B(\omega)$

Neural Network

- We used a three-layer neural network to distinguish between human speech and background noise.
- More specifically, our neural network consists of an input layer, a hidden layer, and an output layer and uses the backpropagation algorithm and gradient descent to minimize the cost function.



System Block Diagram



- BSS separates input signal into two components.
- Neural network places a weight on each output based on how much human speech they contain.
- The system outputs the signal with the highest weight above a threshold.

Results

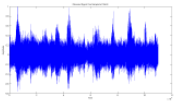


Figure 1: Mixed signal observed at 1st microphone

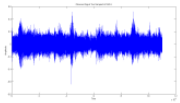


Figure 2: Mixed signal observed at 2nd microphone

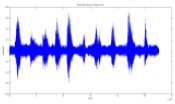


Figure 3: Estimate source signal one – human speech

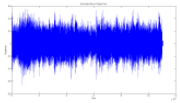


Figure 4: Estimated source signal two – music

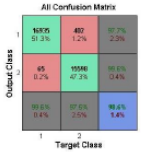


Figure 5: Accuracy of neural network

Parameters	
Sampling Rate	16 kHz
Length of STFT	512
Number of FFT Points	4096
Overlap	492
Number of inputs	2
Permutation Reference Range	5

Figure 6: Input parameters into the system.

Conclusion and Future Steps

- Figures 3 and 4 show that our BSS method is able to separate an input signal into two independent and distinct sources.
- Our neural network can perform binary classification with 98.6% accuracy.
- Implementation in real-time.
- Explore possibility to separate sources from more than two inputs.

References and Acknowledgements

- Hyvärinen, Aapo, and Erkki Oja. "Independent component analysis: algorithms and applications." *Neural networks* 13.4 (2000): 411-430.
- Asano, Futoshi, et al. "Combined approach of array processing and independent component analysis for blind separation of acoustic signals."*Speech and Audio Processing, IEEE Transactions on* 11.3 (2003): 204-215.
- Special thanks to Professor Richard Baraniuk and Dr. Mohammad Golbabaee.

For questions, please contact:
Zichao.Wang@rice.edu; Tianyi.Yao@rice.edu; Stephen.Xia@rice.edu; Shengjia.Zhao@rice.edu

Our Project Poster